Multi-party Specification

Anthony Finkelstein & Hugo Fuks

Imperial College of Science, Technology & Medicine (University of London), Dept. of Computing, 180 Queens Gate, London SW7. acwf@uk.ac.ic.doc

0 Abstract

This paper examines a formal model of how specifications can be constructed from multiple viewpoints and presents some tools to support this approach. The development of specifications is presented as a dialogue in which the viewpoints negotiate, establish responsibilities and cooperatively construct a specification. The model is illustrated by means of some small examples.

Keywords: formal specification, distributed artificial intelligence, dialogue, logic, tool support

1 Introduction

"Specification-in-the-large", that is the development of requirements specifications for systems of substantial complexity and scale, mirrors "programming-in-the-large" in raising a variety of difficulties that lie beyond the clerical problems of handling large amounts of information (Cunningham, Finkelstein et al 1985, Finkelstein & Potts 1987). One such difficulty is that of specification from multiple viewpoints (Niskier 1987). Specification-in-the-large is an activity in which there are many participants - clients, systems analysts, engineers, domain experts and so on. Each has differing perspectives on, and knowledge about, the object system, as well as a variety of skills, roles and so on. In some cases the perspectives may be based on underlying contradictions. To construct a specification the participants must cooperate; that is, contribute to the achievement of a joint understanding.

This contrasts with the approach taken by existing specification schemes, methods and tools which are generally based on specification from a single viewpoint and refined using examples that consolidate this weakness.

Our research objective is to develop a formal understanding of specification from multiple viewpoints so that we can both support the construction of formal specifications and reason about the process of specification itself. We aim to encapsulate cooperative specification development strategies, "replay" these strategies (Wile 1983) and develop appropriate support and coordination tools. To do so we have taken what might be broadly termed an AI approach - we have sought to model the mechanisms which underlie the way people carry out the complex task of specification.

2 Dialogue

Our model of specification from multiple viewpoints treats the development of a specification as a dialogue in which the viewpoints negotiate, establish responsibilities and cooperatively construct an overall specification. The term dialogue, as generally used, refers to a conversation or spoken interaction between two or more partners. This definition is suitable as a starting point. Our approach is related to the more restricted (and informal) metaphor of the contract model of specification, exploited in the ISTAR environment (Lehman 1985), in which tasks are shared through the negotiation and award of "contracts" among developers.

Before introducing our model in detail it may be useful to examine the intuitions that suggested this approach.

The most straightforward of these is that it directly mirrors the conventional setting of requirements specification in which clients and systems analysts sit around a table - the clients explaining the requirements, waving documents in the air and occasionally arguing among themselves while the developers ask guiding questions, seek clarification, point out inconsistencies and raise unanticipated consequences.

Examining the way in which complex specifications are built and documented - in natural language - is a well understood way of developing specification techniques with higher expressiveness (Balzer, Goldman & Wile 1978). The approach is exemplified by Gist (Balzer 1985) and Pure Tell (Horai, Saeki & Enomoto 1987). It is not such a great leap of the imagination to extrapolate from this to using the structure of dialogue as an overall setting.

Less directly we regard formal software development as making interpretations between theories (Maibaum, Veloso & Sadler 1985). This process of interpretation is, we suggest, dialogic in form. A formal account of interpretation based on dialogue is developed in Niskier, Fuks & Sadler (1988).

Finally it is hoped that by having a model which is based on dialogue we might have a convenient framework to understand empirical studies of specification, generally in the form of protocols, which are notoriously difficult to analyse (Soloway 1986).

3 Model

The model we present has two parts: an underlying *viewpoint architecture* and a *dialogue scheme* animated by that architecture. Our model also comes in two basic flavours. Two party dialogues (or multi-party dialogues consisting of many two party dialogues) and true N party (N>2) dialogues. Currently we have a detailed understanding of two party dialogues, which we will illustrate in this paper, and have established the formal underpinning for N party dialogue which will be briefly reviewed.

The underlying vehicle for defining our model is a formal account of dialogue. Such accounts have their roots in a number of different traditions:

the *game theoretic semantics tradition* in which dialogue "games" are used to define the meanings of components of a formal language, for example Lorenz (1982);

the *foundations of logic tradition* in which an understanding of the communicative context of argument is examined to understand the development of different logical traditions, notably Hamblin (1987);

the *computer human interaction tradition* in which representations of dialogues are developed for design and evaluation of user interfaces, for example Schneiderman (1982) and Green (1983);

the *rhetorical or argumentative tradition* in which a model of dialogue provides the normative base for deciding what constitutes rhetorical "competence", for example Allwood (1986);

the *natural language processing tradition* in which computationally tractable models are sought to provide a basis for automatically interpreting and generating dialogues, for example Carbonell (1982);

the *distributed artificial intelligence tradition* in which computational models of multi-agent "negotiation" are constructed to integrate diverse knowledge sources, notably Erman & Lesser (1975), Smith (1980), Smith & Davis (1981), Kornfeld & Hewitt (1981) and Lenat (1975).

We have sought to combine the formal apparatus - dialogue logics - of the foundations of logic tradition with the approach - cooperation and negotiation - of the distributed artificial intelligence tradition. We have explicitly rejected the competitive approach typical of the game theoretic semantics tradition and are not directly concerned with the discourse level issues that dominate both the natural language processing and the rhetorical tradition. The descriptive tools provided by the computer human interaction tradition lack the required expressiveness for the less highly constrained dialogues on which we have focussed.

3.1 Viewpoint Architecture

Figure 1 shows a block diagram of the viewpoint architecture. We have shown two main participants (Viewpoints A & B) in the diagram. Each additional participant in a dialogue (such as Viewpoint N) has a similar structure.

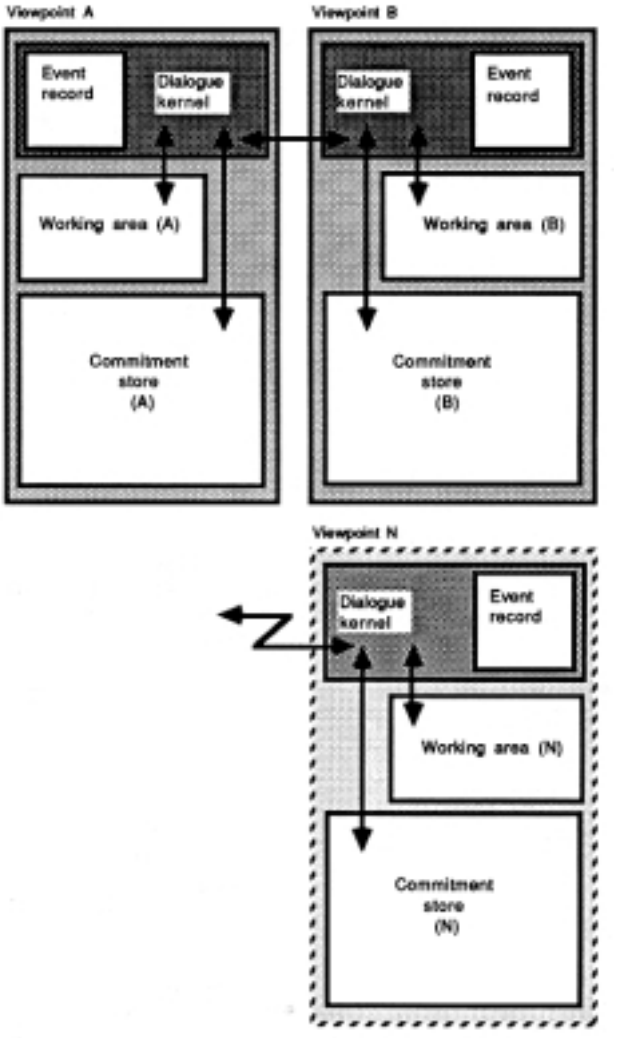

Figure 1 Block diagram of viewpoint architecture

The main building blocks of the viewpoint architecture are as follows:

Viewpoint

A viewpoint is a "logical" participant in the dialogue. We can loosely define a viewpoint as an agent responsible for maintaining a particular perspective. A physical participant in a dialogue may "act the part of" or "present" many logical viewpoints. For example, a librarian might be responsible for both acquisitions policy and disposals. Our model does not as yet include structured groups of viewpoints.

Commitment store

Each viewpoint has a commitment store which holds it's commitments within the dialogue. A commitment is the public engagement to a statement that restricts freedom of action. A commitment to a statement is, in effect, holding yourself out as liable for the consequences of that statement (just as clients in "real-life" software specification "commit" themselves by signing off a requirements statement). As a concept commitment needs to be carefully distinguished from epistemic notions such as belief, which are essentially private

and which we exclude from our model both on philosophical and technical grounds. The general status and role of commitments is discussed by Winograd & Flores (1986). In the context of design Thimbleby (1988) suggests that "... abstraction and commitment are inverse processes - an abstraction is an outcome of relaxing commitments and a representation is the outcome of making commitments".

The contents of the commitment store of each participant changes as the dialogue progresses. A viewpoint can read from any store. The only way it can alter the commitments of any viewpoint, including it's own, is through participation in the dialogue. A specification is the pool of commitments that result from such a dialogue, an approach closely akin to that of the requirements analysis method CORE (Mullery (1985).

Working area

The private "sketch-pad" or database of the viewpoint. It contains the internal or working statements which do not have the full status of public engagement. No other viewpoint can directly access the working area nor is there any obligation for the viewpoint to maintain it's consistency. The working area is an essential element of the architecture but is not treated explicitly in the dialogue scheme outlined below.

Event store

The event store keeps a record of the "dialogue events". It is used by the dialogue kernel to maintain the legality of the dialogue. Dialogues are flexible and dynamic, participants make moves to realise their aims based on the revealed "record of play".

Dialogue kernel

The dialogue kernel is the independent controller of a viewpoint. The role of the dialogue kernel is to implement the common dialogue scheme for each (loosely coupled) viewpoint.

In the two party version of the model responsibility for issues such as control of the domain of discourse, initiation and completion of dialogues are implicitly assigned to the individual viewpoints. An alternative approach is to provide each dialogue with an "agenda" which explicitly handles these issues by establishing a larger set of commitments at the level of the dialogue itself (this can be thought of as analogous to the organising role played by a specification method). We are investigating this approach further in the context of N-party dialogues.

3.2 Dialogue Scheme

Formal schemes that fit with the architecture outlined above and with our overall approach to modelling dialogue have been proposed by Hamblin (1971) and more fully worked by Mackenzie (1981, 1985). We have adopted Mackenzie's dialogue system - DC - and, with some changes and substantial reinterpretation, are using it as the basis for validating our intuitions. Below we present a brief overview of the main elements of DC, using our own notation.

The scheme is presented in terms of three important constructs:

(i) Acts

Equivalent to "locutions", "utterances" or "speech acts". Consist of a statement and a modifier, represented **modifier**(Statement). Statements are constructed in a propositional language which includes negation, conditional and conjunction of statements.

Act modifiers are as follows:

Assertions, to be read as "It is the case that Statement", notationally **asserts**(Statement).

Denials, to be read as "I deny that it is the case that Statement", notationally **denies**(Statement).

Questions, to be read as "Is it the case that Statement?", notationally **questions**(Statement).

Withdrawals, to be read as "No commitment to Statement", notationally **withdraws**(Statement).

Challenges, to be read as "Why is it to be supposed that Statement", notationally **why**(Statement).

Resolution demands, to be read as "Resolve your commitments", notationally **resolve**(CS(Viewpoint)).

(ii) Events

Represented by a triple of the form <Stage, Viewpoint, Act>. Stage marks the progress of the dialogue, stage, stage+1 and so on. Viewpoint indicates the current speaker. A dialogue is a sequence of such events.

(iii) Commitments

Represented **committed**(Stage,Viewpoint).

These constructs are used in the rules of the scheme which are divided into three subsets:

(i) Dialogue rules

Establish the "etiquette" or rules governing the legitimate shape of the interaction, they provide a way of maintaining a "legal" dialogue.

For example:

Dialogue rule Quest (Questions):

After the questioning of a statement (**questions**(Statement)), the next event must be either the assertion (confirmation) of that statement, it's withdrawal or it's denial (**asserts**(Statement), **withdraws**(Statement) or **denies**(Statement)).

No legal dialogue of length stage+1 contains an event <stage-1,hearer,**questions**(Statement)> unless it also contains an event <stage,speaker,**asserts**(Statement)> ∨ <stage,speaker,**withdraws**(Statement)> ∨ <stage,speaker,**denies**(Statement)>.

(ii) Commitment rules

Set out how acts affect the commitment store of each viewpoint (we see these changes to the commitments as more or less equivalent to the "high level edits" described by Feather (1987)).

For example:

Commitment rule W:

After a withdrawal the statement is removed from the speaker's commitment store, the hearer's store remains unchanged.

After <stage,speaker,**withdraws**(Statement)>
committed(stage+1,speaker)=
    committed(stage,speaker) - {Statement}
committed(stage+1,hearer)=
    committed(stage,hearer)

(iii) Argument forms

Define, syntactically, the form of reasoning permissible within the dialogue and common to it's participants. Our presentation of DC primarily involves "modus ponens", though addition of other schemas to fit various logical tastes is a relatively simple matter. The argument form mechanism for modus ponens is embedded in the rule below:

Commitment rule G:

After an assertion (AnotherStatement) which occurs as a reply to a challenge (**why**(Statement)) both views are committed to the reply (AnotherStatement) and to the conditional (AnotherStatement -> Statement).

After <stage,speaker,**asserts**(AnotherStatement)>
where the preceding dialogue event was
<stage-1,speaker,**why**(Statement)>
committed(stage+1,speaker)=**committed**(stage,speaker) ∪
    {AnotherStatement, AnotherStatement -> Statement}
committed(stage+1,hearer)= **committed**(stage,hearer) ∪
    {AnotherStatement, AnotherStatement -> Statement}

As can be seen, not only was (AnotherStatement) added to both stores, as would be expected, but also (AnotherStatement -> Statement). If we take (AnotherStatement, AnotherStatement -> Statement) and apply the modus ponens rule to it, we deduce (Statement) - exactly what was originally challenged.

4 Examples

Our overall approach may be clarified by looking at some examples. We shall use a small case study concerning description of an automated travel ticketing system. In this case study various statements about travel and travel discounts are distributed between the working area of two viewpoints.

Figures 2, 3 & 4 below, differ slightly from Figure 1 for ease and economy of presentation. The figures show two viewpoints called respectively A and B each represented by a shaded box. Each viewpoint has a working area with different contents (WA (A) and WA (B)). Commitments are represented in separate stores (CS (A') and CS (B')). The current dialogue event is given in a box at the top of the diagram alongside an arrow that points from the originator of the event (speaker) to the recipient (hearer). The commitments stores resulting from that dialogue event (CS (A) and CS (B)) are shown at the bottom of the diagram. Other figures just show the commitment stores with the original commitments in a box above a bar showing the dialogue event and the resulting commitments below. These commitments may in turn be altered by a subsequent event.

Figure 2 illustrates the addition of information to a description (Example A). Let us follow what happens in the process of making this addition.

Initially the commitment store is empty. The speaker (B) asserts the statement, in this case *ticket* ["can obtain a ticket"] (an immediate consequence of the content of its working area (*discount_fare*, *discount_fare* -> *ticket*) ["paid a discount fare" and "paying a discount fare implies that you can obtain a ticket]), and so by:

Commitment rule S:

If a statement has been made which is not the reply to a challenge then the speaker and the hearer are obliged to place it in their commitment store.

After <stage,speaker,**asserts**(Statement)>where the preceding dialogue event was not
<stage-1, hearer,**why**(AnotherStatement)>
committed(stage+1,speaker)=
    committed(stage,speaker) ∪ {Statement}
committed(stage+1,hearer)=
    committed(stage,hearer) ∪ {Statement}

The rule S defines an important feature of this dialogue scheme. A viewpoint is committed to anything stated by another viewpoint. This commitment can only be removed by a subsequent withdrawal or challenge.

The resulting commitment stores are CS (A) and CS (B). Both A and B are committed to *ticket* and must answer for any consequences of this commitment and other commitments added in a similar manner. This process of straightforwardly adding commitments, which we can clearly continue, can be termed "simple elaboration". An enhanced description or specification is built up in the commitment stores and shared between the participating viewpoints.

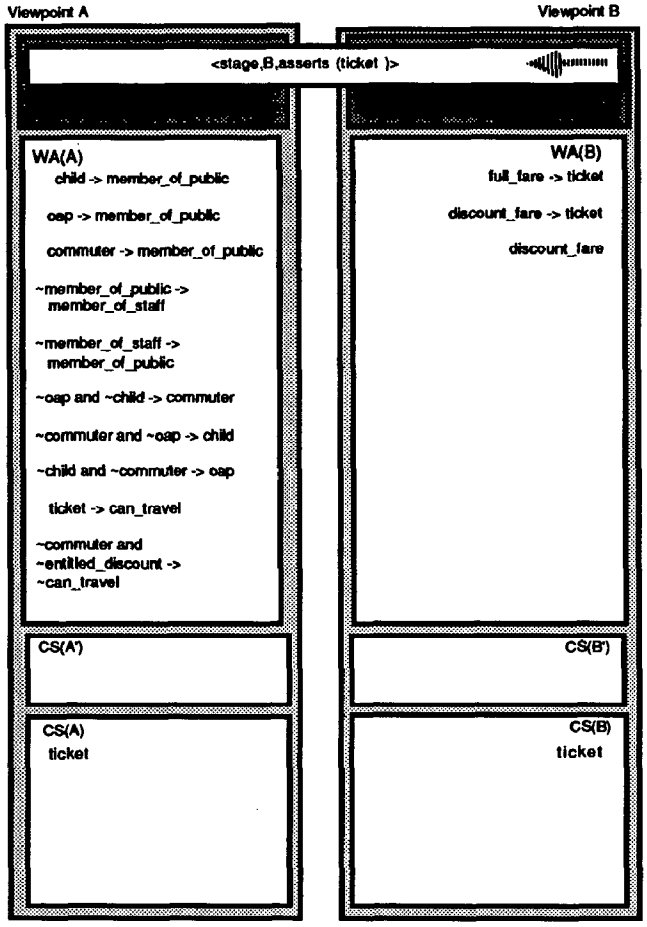

Figure 2 Example A (1 of 1)

Let us now consider a slightly more complicated example (Example B) illustrating refinement of a description. We start at a point some way into a set of dialogues (Figure 3) with CS (A') containing a commitment to *child -> ticket* ["a child can obtain a ticket"]. Viewpoint A, the speaker and initiator of this part of the dialogue, asks something along the lines of "why is it to be supposed that a child can obtain a ticket?".

It should be noted that in this setting asking why is a demand for evidence, not for an explanation. So by:

Commitment rule Y:

After a challenge the hearer adds the challenged statement to it's own commitment store and the speaker removes the statement from it's commitment store, replacing it by the challenge itself. This is necessary to avoid the problem of circularity ("Why is the book on loan?","Because it is out of the library!","Why is it out of the library?","Because it is on loan!!" and so on).

After <stage,speaker,**why**(Statement)>
committed(stage+1,speaker)=
    committed(stage,speaker) -
        {Statement} ∪{**why**(Statement)}
committed(stage+1,hearer)=
    committed(stage,hearer) ∪ {Statement}

Observe that the placing of {Statement} in the hearer's commitment store forces a reaction - either a challenge to "give a good reason" for the statement or a withdrawal in order not to be committed to it. As a last resort the hearer may demand a resolution over the speaker's commitment store, we will illustrate this in a subsequent example.

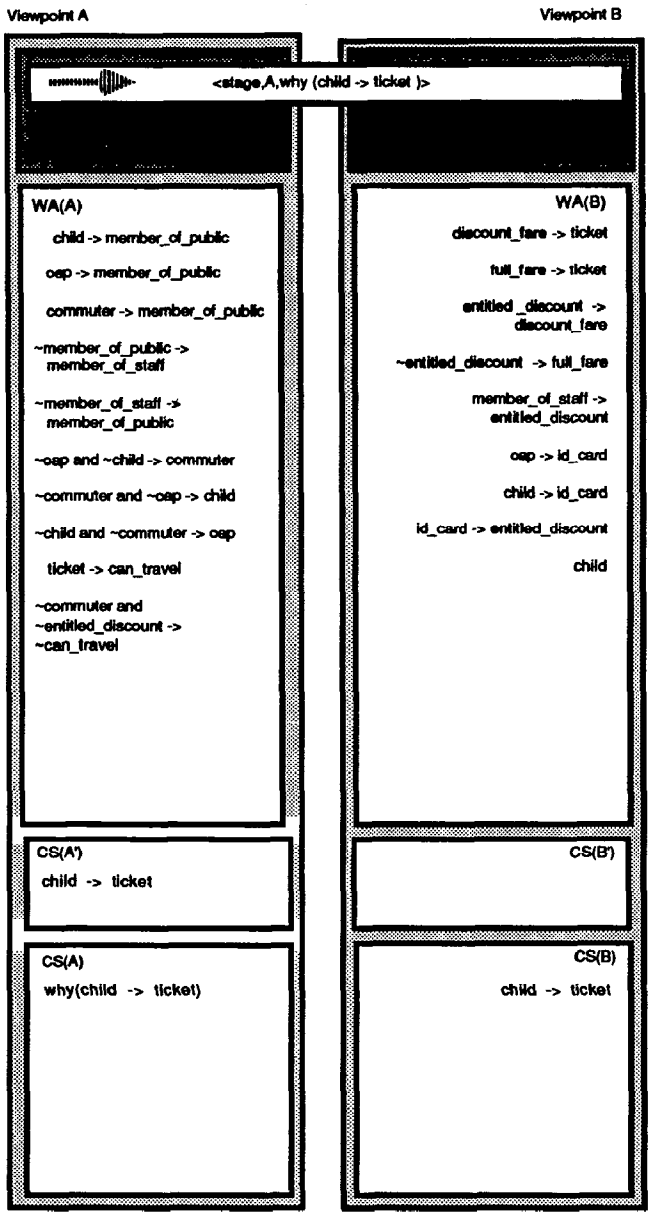

Figure 3 Example B (1 of 5)

CS (A) and CS (B) now form the commitments for the next stage of the dialogue and reappear as the new CS (A') and CS (B') shown in Figure 4. Viewpoint B replies to maintain dialogue legality as indicated by:

Dialogue rule Form:

Each viewpoint contributes an act at a time, in turn each act must be well formed that is a statement, question etc.

No legal dialogue contains an event <stage,given_viewpoint,Act> if it also contains an event <stage-1,given_viewpoint,AnotherAct> or if Act is not properly constructed.

The act that follows is the assertion of *discount_fare* which is taken from the statement contained in the working area that *discount_fare -> ticket*. The resulting commitment stores CS (A) and CS (B) are derived according to the rule below:

Commitment rule G (for the challenge of an implication):

After an assertion (AnotherStatement) which occurs as a reply to a challenge (**why**(Statement1 -> Statement2)) both views are committed to the reply (AnotherStatement) and to the conditional (AnotherStatement -> Statement2).

After <stage,speaker,**asserts**(AnotherStatement)>
where the preceding dialogue event was
<stage-1,speaker,**why**(Statement1 -> Statement2)>
committed(stage+1,speaker)=**committed**(stage,speaker) ∪
{AnotherStatement, AnotherStatement -> Statement2}
committed(stage+1,hearer)= **committed**(stage,hearer) ∪
{AnotherStatement, AnotherStatement -> Statement2}

Figures 5, 6 & 7 show the continuation of this dialogue in which the refinement (all the steps required to show why a child can obtain a ticket) is completed in a constructive manner by a process of "dialogue led" backward chaining.

Note that in Figure 5 Viewpoint A can challenge either *discount_fare -> ticket* or *discount_fare*. It chooses to challenge *discount_fare* because this was the actual reply of B, while *discount_fare -> ticket* is a construct of the commitment rule G.

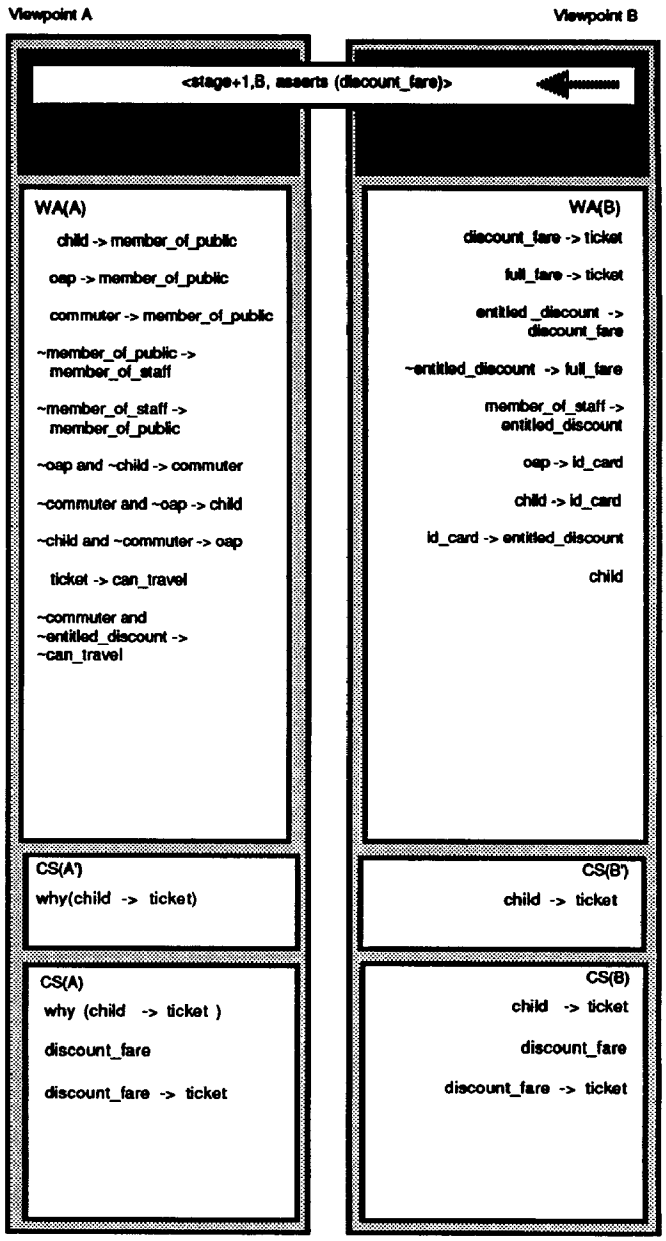

Figure 4 Example B (2 of 5)

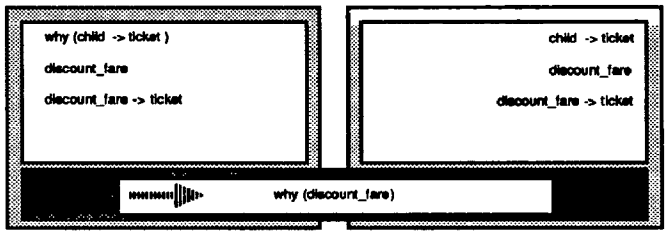

Figure 5 Example B (3 of 5)

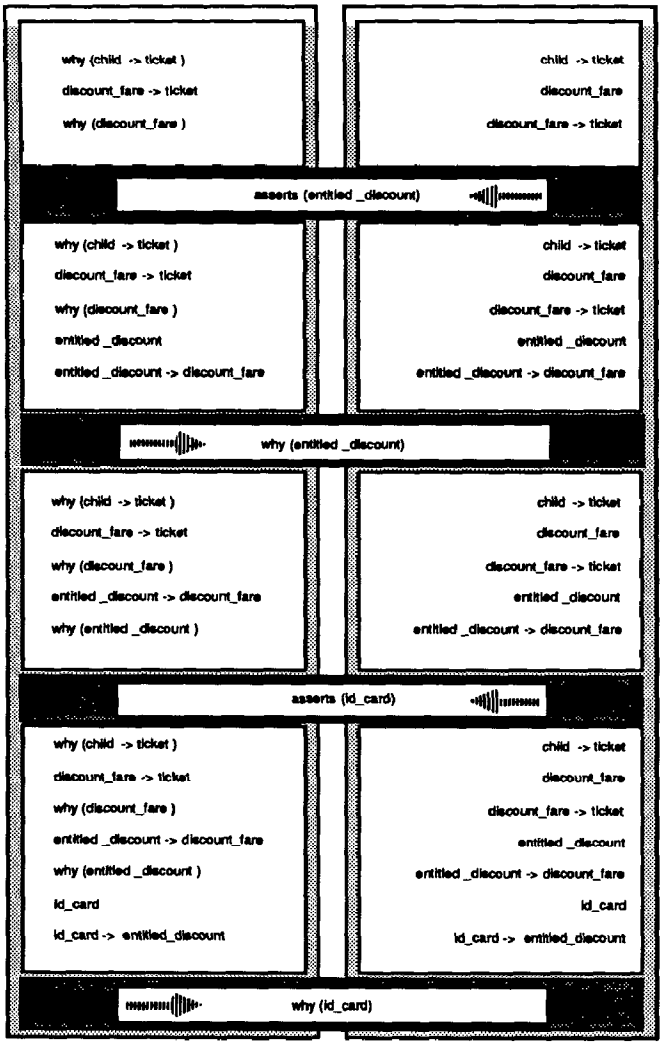

Figure 6 Example B (4 of 5)

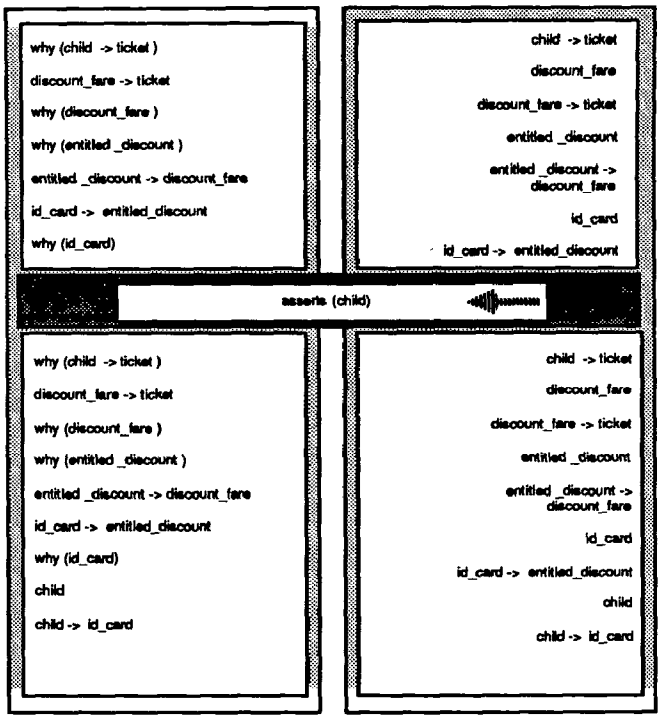

Figure 7 Example B (5 of 5)

The final example (Example C) we will consider is again more complex, illustrating a number of features including progressive verification of one viewpoint with respect to another. By looking at WA(A) in Figure 8 it should be easy to spot the inconsistency which has been introduced (*~id_card*) [" it is not the case that there is an id_card "], as a result of which notice the inconsistency that may possibly arise between Viewpoint B (which is working on the basis of *oap -> entitled_discount* ["oap's are entitled to a discount"] and Viewpoint A.

In Figure 8 Viewpoint B challenges *oap -> entitled_discount*. The resulting commitment stores, CS (A) and CS (B) are derived similarly to the previous examples.

B replies (Figure 9) by asserting *id_card* obtained by matching with the implication *id_card -> entitled_discount* and the commitments are established according to Commitment rule G (for the challenge of an implication) which we have also seen before. B now challenges *id_card* and A withdraws it being unable to deny it due to Dialogue rule Chall which, substantially abbreviated, states:

Dialogue rule Chall (Challenges):

The reply to a challenged statement must be the withdrawal of the statement or it must be the resolution demand of an immediate consequence conditional of the statement whose consequent is the statement and whose antecedant is a conjunction of statements to which the challenger is committed or it must be a statement to which the challenger is not committed.

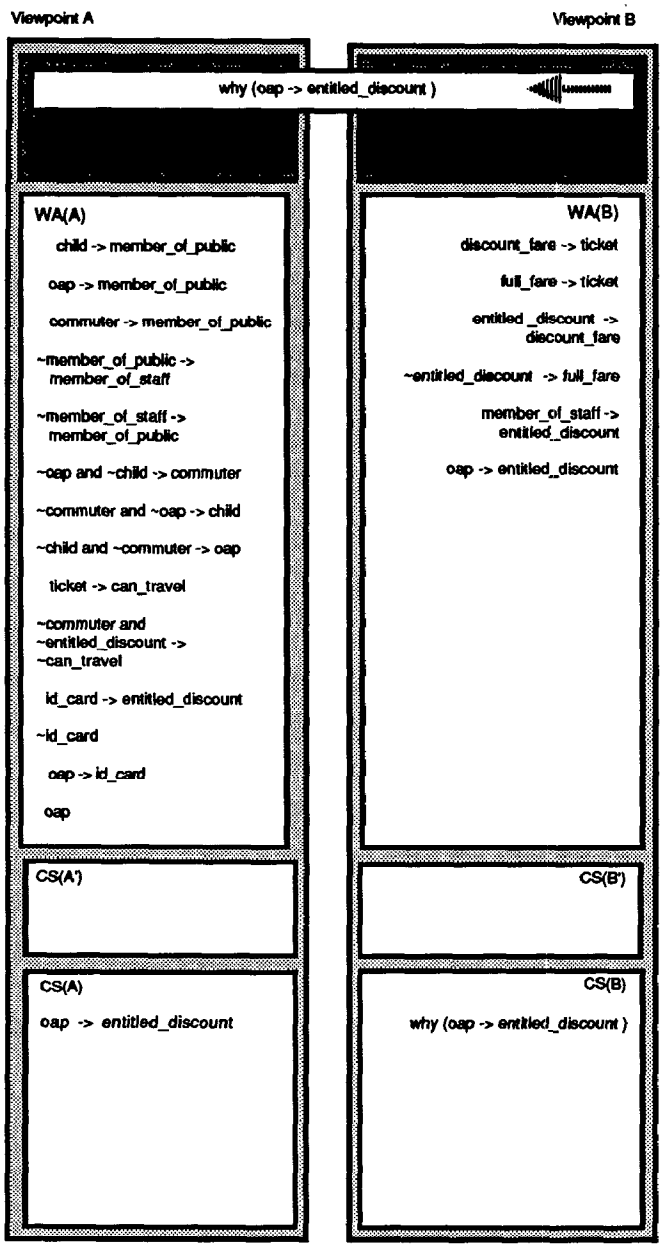

Figure 8 Example C (1 of 5)

Some straightforward question and answer follows with the results determined by Commitment rule S above and:

Commitment rule Q:
Questions do not affect commitment stores.

After <stage,speaker,**questions**(Statement)>
committed(stage+1,speaker)=**committed**(stage,speaker)
committed(stage+1,hearer)= **committed**(stage,hearer)

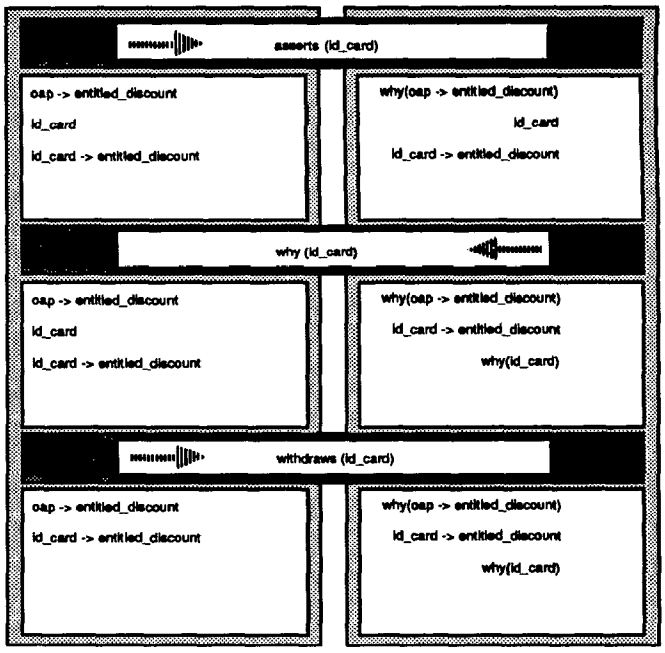

Figure 9 Example C (2 of 5)

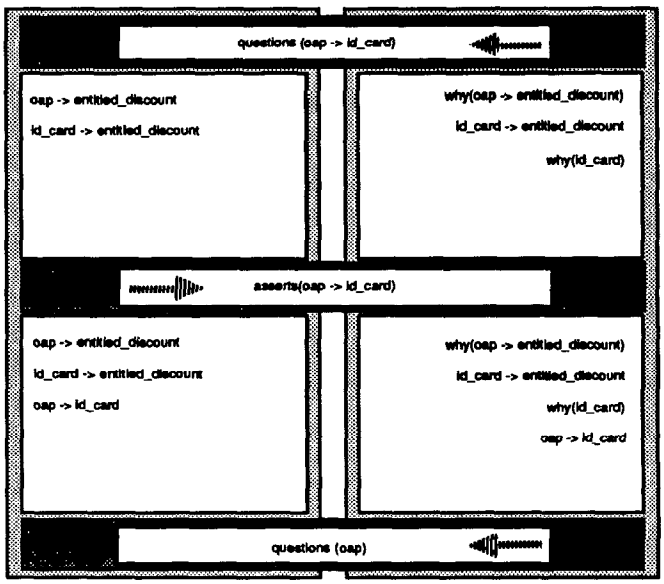

Figure 10 Example (3 of 5)

By Figure 11 Viewpoint B questions *id_card*, which it is free to question because it is not on A's commitment store, and A, as a result of an inconsistency deep in it's working area, denies it with the resulting commitments derived by Commitment rule D:

Commitment rule D:

If a denial of a statement has been made then the speaker and the hearer are obliged to place the negation of the statement in their commitment store.

After <stage,speaker,**denies**(Statement)>
committed(stage+1,speaker)=

committed(stage,speaker) ∪ {~Statement}
committed(stage+1,hearer)=

committed(stage,hearer) ∪ {~Statement}

B immediately demands that A, having denied an immediate consequence of it's commitments resolve it's commitment store, which is now inconsistent (using modus ponens we have *(oap,oap ->id_card) —> id_card* which is of course inconsistent with *~id_card*):

Commitment rule R:

Resolution demands do not affect commitment.

After <stage,speaker,**resolve**(CS(hearer))>
committed(stage+1,speaker)=**committed**(stage,speaker)
committed(stage+1,hearer)=**committed**(stage,hearer)

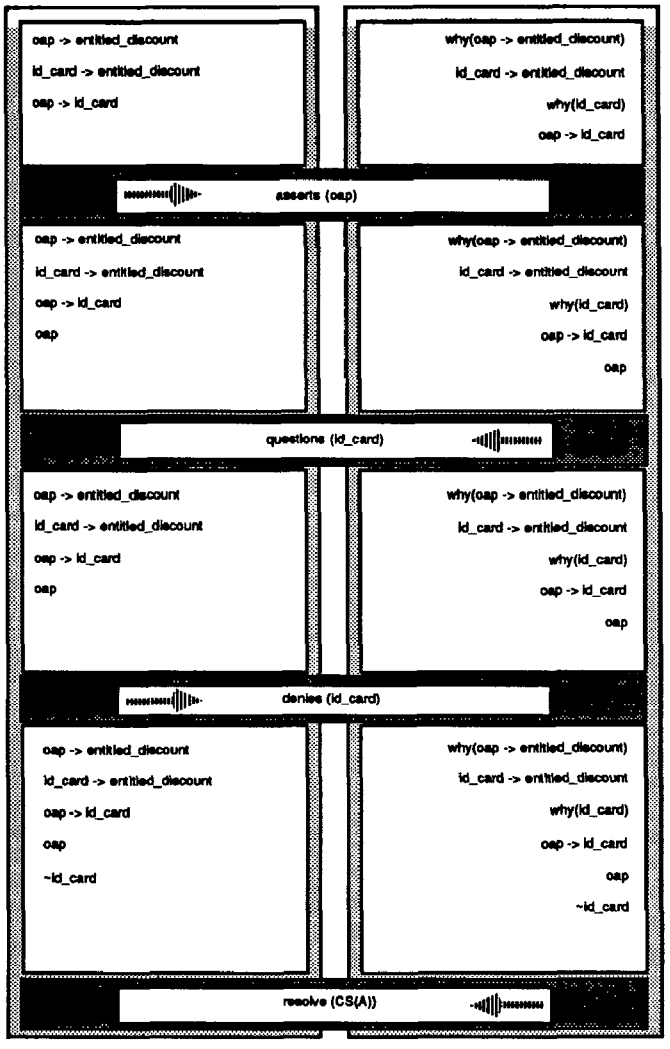

Figure 11 Example C (4 of 5)

In Figure 12 Viewpoint A withdraws it's previous denial (it is constrained to do so by the dialogue rules) restoring consistency by adjusting the commitments according to Commitment rule W given in our overview of the dialogue scheme.

B follows suit by also withdrawing the inconsistency, which if not removed would now leave it liable to a resolution demand from A, and so the dialogue concludes with a shared description and discovery of the "misunderstanding" hidden in A's working area.

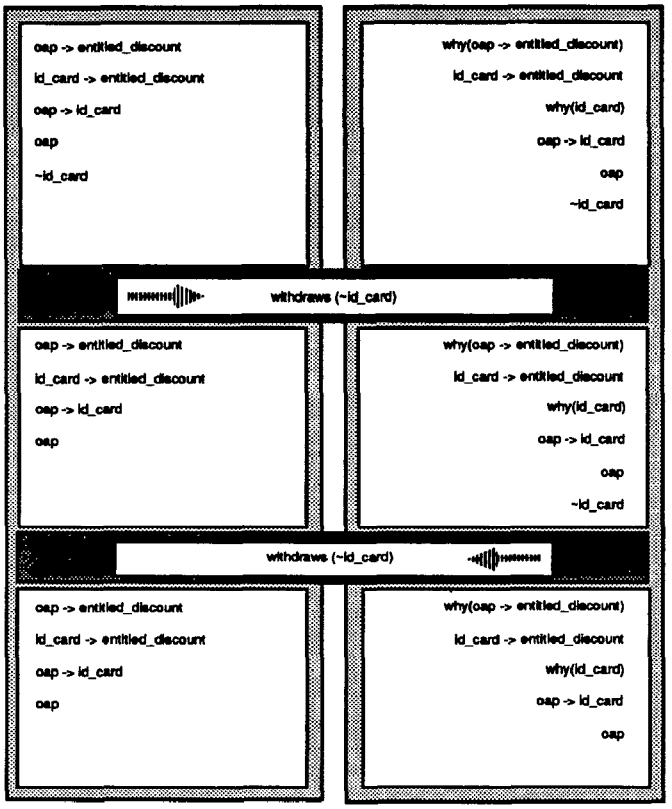

Figure 12 Example C (5 of 5)

5 N-party

An outline of the features necessary to extend the DC dialogue scheme to N-party (N>2) has been developed. The primary element of this extension is the addition of a typed interface to viewpoints that filters commitents. This addition introduces the idea of an audience and substantially changes the way in which the progress of the dialogue is marked, a change which in turn necessitates changes in the basic constructs of our scheme. Dialogue acts are grouped into those which are directed, that is intended to elicit a response from a specified viewpoint, and those which are general, to which any viewpoint can respond. Changes in the form of the two party DC rules are required to accommodate the enhancements notably additional commitment rules to prevent circularity in argument. A number of small examples have been developed and investigated using the outline scheme however there still remain difficulties that relate to the coordination of viewpoints and conventional distributed processing problems such as fairness. The agenda, briefly

discussed earlier, provides a structure within which these issues might be resolved, this is the subject of further work.

6 Automated support

Our approach to specification development clearly requires automated support. Such support serves two purposes: to act as a workbench while developing an improved understanding of the model and enhancing dialogue schemes (without support even relatively simple examples are awkward to handle); to be used as the core of a specification support environment based on the principles we have outlined. In such an environment replaying a "development history" would be equivalent to running through a record of dialogue events (this can be compared to the approach of Conklin 1989). In principle it would also be possible to record and replay generalised dialogue "strategies" (an elaboration strategy, verification strategy and so on).

We have developed two dialogue support systems (IC-DC One & IC-DC Two) which animate, albeit in a simple minded way, the dialogue scheme. These tools allow the user to develop simple dialogues like the examples above and then replay them in whole or in part. IC-DC One is written in Prolog and has been used to help us to understand and enhance the dialogue rules. IC-DC Two is written in Smalltalk-80 and has been used to investigate an appropriate architecture for a specification support environment and N-party extensions to the model.

In both tools the dialogues are monitored for legality and illegal dialogues can be explained and rolled back to a legal state. Users may view the commitment stores of the participating views and may change the course of the dialogue by editing the commitment stores directly.

7 Conclusions

We have presented dialogue as a basis for constructing specifications from multiple viewpoints. This approach combines an intuitively appealing model with a non-classical formal framework. We have developed a detailed understanding of, and automated support for, cooperation and negotiation of two viewpoints and laid the ground work for N-party cooperation. The model has been validated by experience on a large number of small examples. Our approach links work on specification with advances in the foundations of logic, linguistic philosophy, distributed artificial intelligence (in which area we believe our model makes some contribution) and the use of social metaphors in computing.

Having made a radical departure from existing models it should stressed that there remains a substantial amount of foundational work to be done to make extended dialogue models which are both computationally tractable and formally sound. To this end we are currently engaged in the construction of a dialogic framework for theorem proving (Fuks, Pequeno & Sadler 1988). In this work we are developing an idea posed by Hintikka (1973), that the act of proving a theorem can be seen as a dialogue between "nature" and the logician.

It is important to emphasise that our model is very sparse. By basing our work on a formal model of argumentation there are practical limitations in both the underlying language and the dialogic strategies we can capture. Observational studies of specification construction (Fickas, Collins & Olivier 1987) show clearly the sophisticated strategies, such as example generation, which are employed during this activity. We aim, within our overall framework, to be able to capture such strategies but in doing so we must of necessity make the delicate balance between this concern and the formal properties of our model. Our argument is not that the strategies we have succeeded in capturing are sufficient in themselves for understanding specification but rather that our approach provides a foundation on which such an understanding may be built.

Our immediate aim is to continue work revising and extending the dialogue schemes, including generalising dialogue strategies, with the long term objective of developing a full specification support environment based on the approach we have outlined. Our vision of the future sees dialogue as providing the logical equivalent of Unix-style pipes and filters to support the communication of complex formal objects between distributed and cooperating communities of tools and users.

Acknowledgements

The authors would like to thank their colleagues and students most notably Wayne Butcher who has been responsible for implementing IC-DC Two, also Martin Sadler and Celso Niskier for the lively critical discussion which has contributed significantly to the work this paper reports. Hugo Fuks is supported by the Brazilian National Research Council CNPq, grant 202471/86-cc.

References

Allwood, J. (1986); Logic and Spoken Interaction; In: Myers,Brown & McGonigle (Eds), Reasoning and Discourse Processes, Academic Press Cognitive Science Series, pp 67-94.

Balzer, R. (1985); A 15 Year Perspective on Automatic Programming; IEEE Trans. Software Engineering; SE-11,11,pp 1257-1267.

Balzer, R. Goldman, N. & Wile, D. (1978); Informality in Program Specifications; IEEE Trans. Software Engineering; SE-4,2, pp 94-103.

Carbonell, J.G. (1982); Meta-Language Utterances in Purposive Discourse; Carnegie-Mellon University Tech. Report, CMU-CS-82-125.

Conklin, J. (1989); Design Rationale and Maintainability; Proc. 22nd Hawaii International Conference on Systems Sciences, V2, pp 533-539, IEEE CS Press.

Cunningham, J. Finkelstein, A. Goldsack, S. Maibaum, T. & Potts, C. (1985); Formal Requirements Specification - The Forest Project; Proc. 3rd IWSSD; pp 186-191, IEEE CS Press.

Erman, L. & Lesser, V. (1975); A Multi-Level Organization for Problem Solving Using Many, Diverse, Cooperating Sources of Knowledge; Proc. IJCAI-75, pp 483-489.

Feather, M. (1987); Constructing Specifications by Combining Parallel Elaborations; To appear IEEE Trans. Software Engineering.

Fickas, S. Collins, S. & Olivier, S. (1987); Problem Acquisition in Software Analysis: a preliminary study; University of Oregon Tech. Report, CIS-TR-87-15.

Finkelstein, A. & Potts, C. (1987); Building Formal Specifications Using "Structured Common Sense"; Proc. 4th IWSSD; IEEE CS Press.

Fuks, H. Pequeno, M. & Sadler, M.(1988); A Dialogic Framework for Theorem Proving, Imperial College, Department of Computing Tech. Report, DoC 88/4.

Green, M. (1983); Report on Dialogue Specification tools; In: Pfaff (Ed) Proc. Wrkshp. User Interface Management Systems 1983; Springer-Verlag.

Hamblin, C. (1971); Mathematical Models of Dialogue; Theoria, V2, pp130-155.

Hamblin, C (1987); Imperatives; Basil Blackwell, Oxford.

Hintikka, J (1973); Logic, Language-Games and Information; Clarendon Press, Oxford.

Horai,H. Saeki, M. & Enomoto, H. (1987); Specification Based Software Development System Pure Tell, IIAS Research Report No. 73.

Kornfeld, W. & Hewitt, C. (1981); The Scientific Community Metaphor; IEEE Trans Systems, Man & Cybernetics, SMC-11,1, pp 24-32.

Lehman, M. (1985); Approach to a Disciplined Development Process - The ISTAR Integrated Project Support Environment; Imperial College Dept. of Computing Tech Report 85/19.

Lenat, D. (1975); Beings: Knowledge as Interacting experts; Proc. IJCAI-75, pp126-133.

Lorenz, K. (1982); On the Criteria for the choice of Rules of Dialogic Logic; In Barth & Martens (Eds), Argumentation: approaches to theory formation, SLCS V8, John Benjamins, Amsterdam.

Mackenzie, J. (1981); The Dialectics of Logic; Logique et Analyse, V24, pp 159-177.

Mackenzie, J. (1985); No Logic before Friday; Synthese, V63, pp 329-341.

Maibaum,T., Veloso, P. & Sadler, M. (1985); A Theory of Abstract Data Types for Formal Development: bridging the gap?; Proc. Colloq. Software Engineering Berlin 1985; LNCS 186, Springer Verlag.

Mullery, G. (1985); Acquisition — Environment; In Paul, M. & Siegert, H. (Eds), Distributed Systems: methods and tools for specification, LNCS 190, Springer Verlag.

Niskier, C. (1987); Using Multiple Views in Software Specification; Proc. 5th Workshop on Formal Specification of Abstract Data Types, Gullane, Scotland.

Niskier, C., Fuks, H. & Sadler, M. (1988); Changing Views in Software Specification: interpretation between theories using dialogue; (In preparation).

Schneiderman, B. (1982); Multiparty Grammars and Related Features for Defining Interactive Systems; IEEE Trans Systems, Man & Cybernetics, SMC-12,1, pp 148-154.

Smith, R. (1980); The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver; IEEE Trans. Computers, C-29,12, pp1104-1113.

Smith, R. & Davis, R. (1981); Frameworks for Cooperation in Distributed Problem Solving; IEEE Trans Systems, Man & Cybernetics, SMC-11,1, pp 61-69.

Soloway, E. (1986); Meeting the Challenge of Programming-in-the-Large, 1st Workshop on Empirical Studies of Programmers 1986, pp 263-268; Ablex Publishing Corp.

Thimbleby, H. (1988); Delaying Commitment; IEEE Software; V5 N3, pp 78-86.

Wile D. (1983); Program Developments: Formal Explanations of Implementations; CACM 26(11), pp 902-911.

Winograd, T. & Flores, F. (1986); Understanding Computers and Cognition: A New Foundation For Design; Ablex Publishing Corp.